1998

On a Stochastic Knapsack Problem and Generalizations

Morton, D.P.

5 ON A STOCHASTIC KNAPSACK PROBLEM AND GENERALIZATIONS

Appears is *Advances in Computational and Stochastic Optimization, Logic Programming and Heuristic Search*, ed. Woodruff, D.L., 1998.

David P. Morton[1] and R. Kevin Wood[2]

[1]Graduate Program in Operations Research
The University of Texas at Austin
Austin, TX 78712

[2]Operations Research Department
Naval Postgraduate School
Monterey, CA 93943

Abstract: We consider an integer stochastic knapsack problem (SKP) where the weight of each item is deterministic, but the vector of returns for the items is random with known distribution. The objective is to maximize the probability that a total return threshold is met or exceeded. We study several solution approaches. Exact procedures, based on dynamic programming (DP) and integer programming (IP), are developed for returns that are independent normal random variables with integral means and variances. Computation indicates that the DP is significantly faster the most efficient algorithm to date. The IP is less efficient, but is applicable to more general stochastic IPs with independent normal returns. We also develop a Monte Carlo approximation procedure to solve SKPs with general distributions on the random returns. This method utilizes upper- and lower-bound estimators on the true optimal solution value

in order to construct a confidence interval on the optimality gap of a candidate solution.

1 INTRODUCTION

Consider the following stochastic integer programming problem with random objective function,

$$\begin{array}{lll} \max\limits_{\mathbf{x}} & P\left(\mathbf{rx} \geq c\right) & \\ \text{s.t.} & A\mathbf{x} & \leq \mathbf{b} \\ & \mathbf{x} & \in Z_+^K, \end{array} \tag{1}$$

where Z_+^K is the set of non-negative integer K-vectors, $A\mathbf{x} \leq \mathbf{b}$ are deterministic constraints, c is a deterministic "return threshold" but $\mathbf{r} = (r_1, r_2, \ldots, r_K)$ is a random vector with known distribution. The problem is to select an optimal $\mathbf{x}$, denoted $\mathbf{x}^*$, which maximizes the probability that the return $\mathbf{rx}$ meets or exceeds threshold c.

The stochastic knapsack problem (SKP) is a special case of (1) that may be formulated as follows:

$$\begin{array}{lll} \max\limits_{\mathbf{x}} & P\left(\sum\limits_{k=1}^{K}\sum\limits_{l\in\mathcal{L}_k} r_{kl}x_{kl} \geq c\right) & \\ \text{s.t.} & \sum\limits_{k=1}^{K}\sum\limits_{l\in\mathcal{L}_k} w_k x_{kl} & \leq W \\ & x_{kl} & \in \{0,1\} \quad \forall\, k, l \in \mathcal{L}_k. \end{array} \tag{2}$$

Here, $\sum_{l\in\mathcal{L}_k} x_{kl}$ is the number of items of type k to include in the knapsack, and $|\mathcal{L}_k|$ is an upper bound on this value. The deterministic weight of each item is $w_k > 0$ and W is the known weight capacity of the knapsack. The returns $r_{k1}, \ldots, r_{k|\mathcal{L}_k|}$ for a specific item type k are identically distributed.

The dependence structure of the returns r_{kl} is clearly an important modeling consideration. The variants of the integer SKP addressed in Steinberg and Parks [24], Sniedovich [23], Henig [11], and Carraway *et al.* [4] have returns that are normal random variables which are independent both between item types and within an item type. Independence within an item type means that $r_{k1}, \ldots, r_{k|\mathcal{L}_k|}$ are mutually independent random variables for each k. In some systems this assumption is reasonable: For example, if we are purchasing production equipment in an attempt to satisfy a certain threshold production level and if machines fail independently, it may be appropriate to model the production rates of multiple machines of the same type as independent random variables. On the other hand, realizations of the returns on multiple financial

instruments (e.g., stocks, bonds) of the same type are typically identical. In this latter case, and under the assumption that $|\mathcal{L}_k|$ is limited only by the weight capacity of the knapsack, (2) can be simplified to

$$\begin{aligned} \max_{\mathbf{x}} \quad & P\left(\sum_{k=1}^{K} r_k x_k \ \geq c\right) \\ \text{s.t.} \quad & \sum_{k=1}^{K} w_k x_k \ \leq \ W \\ & x_k \ \in \ Z_+ \quad \forall\, k \end{aligned} \tag{3}$$

where $r_k \equiv r_{k1} = r_{k2} = \cdots = r_{k|\mathcal{L}_k|}$, wp1.

Sniedovich [23] and Henig [11] discuss various optimality criteria for integer SKPs, and Prékopa [20, pp. 243-247] describes methods of handling random objective functions in stochastic programs. Under the assumption of normally distributed coefficients, Greenberg [10], Ishii and Nishida [12], and Morita *et al.* [16] examine SKPs with continuous decision variables. There is a separate literature regarding *on-line* stochastic knapsack problems which have applications in telecommunications; see, for example, Chiu *et al.* [5], Gavious and Rosberg [8], Marchetti-Spaccamela and Vercellis [15], Papastavrou *et al.* [19], Ross [21], and Ross and Tsang [22]. While there are many variants of on-line SKPs, all have the property that items arrive over time and must be accepted or rejected upon arrival without knowing what items will be available for consideration in the future. In this paper we restrict our attention to (2), a "static" SKP.

In Section 2, we discuss the special case of the SKP in which the returns are normal random variables that are independent both between and within item types, i.e., model (2) with returns r_{kl} being mutually independent for all l and k. The returns within a type are identically distributed and are assumed to have integral mean $\mu_k \equiv E r_{k1}$ and integral variance $v_k \equiv \operatorname{var} r_{k1} > 0$. Section 2 derives a simple dynamic-programming-based algorithm for this problem, demonstrates the algorithm's computational effectiveness, and then proposes and illustrates the viability of integer programming methods for solving both the SKP and model (1) which may have general linear constraints. (In the rest of the paper, "DP" will mean "dynamic program" or "dynamic programming," and "IP" will mean "integer program" or "integer programming.") In Section 3, we consider the case where the returns are governed by general distributions that can have arbitrary dependency structures both between and within item types. For such problems, we apply a Monte Carlo procedure that finds a feasible candidate solution $\hat{\mathbf{x}}$ and constructs confidence intervals on its optimality gap, $P\left(\mathbf{r}\mathbf{x}^* \ \geq c\right) - P\left(\mathbf{r}\hat{\mathbf{x}} \ \geq c\right)$.

2 SKP WITH INDEPENDENT NORMAL RETURN DISTRIBUTIONS

Let $r_{kl} \sim N(\mu_k, v_k)$, where $N(\mu_k, v_k)$ is a normal random variable with integral mean μ_k and integral variance v_k, and assume that all r_{kl}, $k = 1, \ldots, K$, $l \in \mathcal{L}_k$, are independent, i.e., the returns are independent both between and within item types. Let $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_K)$ and $\mathbf{v} = (v_1, \ldots, v_K)$. Under these assumptions,

$$\begin{aligned} P\left[\sum_{k=1}^{K}\sum_{l\in\mathcal{L}_k} r_{kl}x_{kl} \geq c\right] &= P\left[N(0,1) \geq \frac{c - \sum_{k=1}^{K}\sum_{l\in\mathcal{L}_k}\mu_k x_{kl}}{\left(\sum_{k=1}^{K}\sum_{l\in\mathcal{L}_k} v_k x_{kl}\right)^{1/2}}\right] \\ &= P\left[N(0,1) \geq (c - \boldsymbol{\mu}\mathbf{x})/\sqrt{\mathbf{v}\mathbf{x}}\right], \end{aligned}$$

where $x_k \equiv \sum_{l\in\mathcal{L}_k} x_{kl}$ and $\mathbf{x} = (x_1, \ldots, x_K)^T \neq \mathbf{0}$. We can therefore maximize the probability of exceeding the return threshold, subject to $\mathbf{x} \in X = \{\mathbf{x} : A\mathbf{x} \geq \mathbf{b}, \mathbf{x} \in Z_+^K\}$, by solving

$$\begin{aligned} \rho^* = \quad & \min_{\mathbf{x}} \quad (c - \boldsymbol{\mu}\mathbf{x})/\sqrt{\mathbf{v}\mathbf{x}} \\ & \text{s.t.} \quad \mathbf{x} \in X \end{aligned} \tag{4}$$

provided $\mathbf{x}^* \neq \mathbf{0}$. This condition is assumed to hold throughout this section since the possibility that $\mathbf{x}^* = \mathbf{0}$ is a simple special case to check. For the stochastic knapsack problem with normal returns, (4) specializes to

SKP(W)

$$\begin{aligned} \rho^*(W) = \quad & \min_{\mathbf{x}} \quad (c - \boldsymbol{\mu}\mathbf{x})/\sqrt{\mathbf{v}\mathbf{x}} \\ & \text{s.t.} \quad \mathbf{w}\mathbf{x} \leq W \\ & \qquad\quad \mathbf{x} \in Z_+^K. \end{aligned} \tag{5}$$

A standard way of attacking (4) and (5), e.g., Henig [11], due in concept to Geoffrion [9], involves solving $\min_{\mathbf{x}\in X}(\lambda\boldsymbol{\mu} + (1-\lambda)\mathbf{v})\mathbf{x}$ multiple times for different values of λ between 0 and 1. However, the method is not guaranteed to achieve an optimal solution when $\rho^* > 0$, i.e., when $P(\mathbf{r}\mathbf{x}^* \geq c) < 1/2$ [11]. Carraway *et al.* [4] use another solution for SKP(W), one that is based on "generalized dynamic programming" [2]. Generalized DP maintains a set of partial solutions for each state of the knapsack (amount of capacity consumed): These partial solutions are ones that might be extended to an optimal solution. (Standard DP maintains only a single solution for each state.) The generalized technique requires that specialized bounds be computed to eliminate partial solutions by proving that they cannot be extended to an optimal solution. In Section 2.1, we develop a DP procedure for solving SKP(W) that is much simpler in concept than the methods described above and is guaranteed to yield

an optimal solution in all cases. In Section 2.2 we show how IP techniques may be used to solve SKP(W) and the more general problem (4). While the IP approach is less efficient than the DP procedure, it can still be used to solve SKP(W) effectively and it has the advantage that any type of linear constraints can be incorporated in the model.

2.1 Dynamic Programming Method

Suppose that we know valid, integral lower and upper bounds, $\underline{v}$ and $\bar{v}$ respectively, on $v^* = \mathbf{v}\mathbf{x}^*$ where $\mathbf{x}^*$ is an optimal solution to SKP(W). Let $V = \{\underline{v}, \underline{v}+1, \ldots, \bar{v}\}$. Since all data are integral, SKP(W) and the following problem are equivalent:

$$\begin{array}{rlll} \rho^* = & \min\limits_{v \in V} \min\limits_{\mathbf{x}} & (c - \boldsymbol{\mu}\mathbf{x})/\sqrt{v} & \\ & \text{s.t.} & \mathbf{v}\mathbf{x} = v & \\ & & \mathbf{x} \in X. & \end{array} \tag{6}$$

For fixed v, the objective function in (6) is minimized when $\boldsymbol{\mu}\mathbf{x}$ is maximized. Therefore, (6) can be solved by solving

$$\begin{array}{lll} \max\limits_{\mathbf{x}} & \boldsymbol{\mu}\mathbf{x} & \\ \text{s.t.} & \mathbf{v}\mathbf{x} = v & \\ & \mathbf{x} \in X & \end{array} \tag{7}$$

to obtain solutions $\mathbf{x}'_v$ for each $v \in V$. Then, $\rho^* = \min_{v \in V} (c - \boldsymbol{\mu}\mathbf{x}'_v)/\sqrt{v}$, and any solution $\mathbf{x}'_v$, $v \in V$, which satisfies $\rho^* = (c - \boldsymbol{\mu}\mathbf{x}'_v)/\sqrt{v}$ is an optimal solution to (4).

Applying the above methodology to SKP(W), (7) becomes

$$\text{KP}(W, v) \qquad \begin{array}{lll} \max\limits_{\mathbf{x}} & \boldsymbol{\mu}\mathbf{x} & \\ \text{s.t.} & \mathbf{w}\mathbf{x} \leq W & \\ & \mathbf{v}\mathbf{x} = v & \\ & \mathbf{x} \in Z_+^K & \end{array} \tag{8}$$

which is just a two-constraint IP that can be solved with reasonable efficiency by extending the standard DP algorithm for the simple knapsack problem. (A text such as Dreyfus and Law [7, pp. 108-110] describes the basic recursion and algorithm; Weingartner and Ness [26] and Nemhauser and Ullman [18] solve knapsack problems with multiple constraints using DP.) Described below is a

scheme for solving SKP(W), based on solving a family of problems of the form KP(W, v), by DP.

Let $f(w, v)$ denote the optimal solution value to KPE(w, v) which is KP(W, v) except that $\mathbf{wx} \leq W$ is replaced by $\mathbf{wx} = w$. For pairs (w, v) that yield an infeasible problem KPE(w, v), we use the convention that $f(w, v) = -\infty$. The first phase of the following algorithm recursively determines $f(w, v)$ for $w \in \{\underline{w}, \underline{w}+1, \ldots, W\}$, and $v \in \{\underline{v}, \underline{v}+1, \ldots, \bar{v}\}$ where $\underline{w} = \min_k w_k$, $\underline{v} = \min_k v_k$, and $\bar{v} = \max_k \lfloor v_k W / w_k \rfloor$. (The floor operator, $\lfloor \cdot \rfloor$, yields the greatest integer that does not exceed its argument. Tighter bounds on v^* are possible, but these choices of $\underline{v}$ and $\bar{v}$ suffice.) Now, define SKPE(w) as SKP(W) but with the constraint $\mathbf{wx} \leq W$ replaced by $\mathbf{wx} = w$. The second phase of the algorithm determines the optimal objective value $\rho(w)$ to SKPE(w) for each $w \in \{\underline{w}, \underline{w}+1, \ldots, W\}$; all possible values of v are examined to do this, for each value of w. (Values of $w < \underline{w}$ are ignored since $\mathbf{x}^* = \mathbf{0}$ is trivially optimal in such cases.) Finally, the third phase extracts the optimal solution $\mathbf{x}^*(w)$ to SKP(w) for each $w \in \{\underline{w}, \underline{w}+1, \ldots, W\}$. This is simply the the best solution to SKPE(w') over all $w' \in \{\underline{w}, \underline{w}+1, \ldots, w\}$.

Algorithm DPSKP
Input: Integer data for SKP(W) with K item types: $\mathbf{w}$, $\boldsymbol{\mu}$, $\mathbf{v}$, c, $W \geq \min_k w_k$.
Output: Optimal solution $\mathbf{x}^*(w)$ and solution value $\rho^*(w)$ to SKP(w) for all $w \in \{\min_k w_k, \ldots, W\}$.
{

/* Phase 1 */

$\underline{w} \leftarrow \min_k w_k$; $\underline{v} \leftarrow \min_k v_k$; $\bar{v} \leftarrow \max_k \lfloor v_k W / w_k \rfloor$;

$f(w, v) \leftarrow -\infty \; \forall \; (w, v)$ with $\underline{w} - \max_k w_k \leq w \leq W$, $\underline{v} - \max_k v_k \leq v \leq \bar{v}$;

$f(0, 0) \leftarrow 0$;

For ($w = \underline{w}$ to W and $v = \underline{v}$ to $\bar{v}$) {

$k(w, v) \leftarrow \text{argmax}_{k \in \{1, \ldots, K\}} [f(w - w_k, v - v_k) + \mu_k]$;

$f(w, v) \leftarrow f(w - w_{k(w,v)}, v - v_{k(w,v)}) + \mu_{k(w,v)}$;

}

/* Phase 2 */

For ($w = \underline{w}$ to W) {

$v' \leftarrow \text{argmin}_{v \in \{\underline{v}, \ldots, \bar{v}\}} (c - f(w, v)) / \sqrt{v}$;

$\rho(w) \leftarrow (c - f(w, v')) / \sqrt{v'}$; $k(w) \leftarrow k(w, v')$;

```
        ŵ(w) ← argmin_{w'∈{w̲,...,w}} ρ(w');
    }
    /* Phase 3 */
    For (w = w̲ to W) {
        x ← 0;   ŵ ← ŵ(w); }
        While (ŵ ≠ 0) { x_{k(ŵ)} ← x_{k(ŵ)} + 1;   ŵ ← ŵ − w_{k(ŵ)}; }
        Print{"Solution to SKP(w) for w=",w,"is x*(w) =",x};
        Print{"with optimal objective value ρ*(w) =",ρ(ŵ(w))};
    }
}
```

To test the algorithm, the data from Steinberg and Parks [24] is used to create 28 SKPs, one for each $W \in \{3, \ldots, 30\}$, and we compare our results against the most recent computational work on these SKPs in Carraway *et al.* [4]. The data describe a small stochastic knapsack problem with $c = 30$ and ten items with weights, means, and variances in the following ranges: $3 \leq w_k \leq 12$, $4 \leq \mu_k \leq 16$, and $8 \leq v_k \leq 25$. DPSKP was programmed in Turbo-Pascal as in [4] but run on a faster personal computer, a Dell Latitude Xpi laptop computer with 40 megabytes of RAM and a 133 MHz Pentium processor. A modest number of enhancements are made in the algorithm for efficiency's sake. For instance, $\bar{v}$ is made a function of w via $\bar{v}(w) = \max_k \lfloor v_k w / w_k \rfloor$. The total solution time for the algorithm (for all values of W between 3 and 30) is 0.026 seconds, which includes printing the solution but excludes time necessary for input. This compares to a solution time (on an IBM PS/2 Model 50) of 114.15 seconds reported in [4] for all 28 problems and a solution time of 14.11 seconds for the single hardest problem ($W = 30$). (The method of [4], although partially based on DP, does not solve SKP(w) sequentially for increasing values of w. Thus, we report the sum of their solution times for all $W \in \{3, \ldots, 30\}$ as well as the time for $W = 30$.)

Solution times for the Steinberg-Parks data can be reduced by taking advantage of the fact that $\bar{v}$ is large compared to $\bar{\mu}$, an analogous integral upper bound on $\boldsymbol{\mu}\mathbf{x}^*$. Let $\underline{\mu}$ be a lower bound on $\boldsymbol{\mu}\mathbf{x}^*$ and let $U = \{\underline{\mu}, \underline{\mu} + 1, \ldots, \bar{\mu}\}$. The optimization of SKP(W) can then be rearranged to

$$\begin{aligned} \rho^* = \min_{\mu \in U} \min_{\mathbf{x}} \quad & (c - \mu)/\sqrt{\mathbf{v}\mathbf{x}} \\ \text{s.t.} \quad & \boldsymbol{\mu}\mathbf{x} = \mu \\ & \mathbf{x} \in X. \end{aligned} \tag{9}$$

For fixed $\mu > c$, the objective is minimized when $\mathbf{vx}$ is minimized, but if $\mu < c$, the objective is minimized when $\mathbf{vx}$ is maximized. Thus, there are two cases to handle ($\mu = c$ is a simple special case we ignore): If (9) is feasible for $\mu > c$, we redefine the lower bound as $\underline{\mu} = c + 1$ and for all values of $\mu \in U$, solve

$$\text{MIN}(\mu) \qquad \begin{array}{lrcl} \min\limits_{\mathbf{x}} & \mathbf{vx} & & \\ \text{s.t.} & \boldsymbol{\mu}\mathbf{x} & = & \mu \\ & \mathbf{x} & \in & X \end{array} \tag{10}$$

for $\mathbf{x}'_\mu$. Otherwise, we redefine the upper bound as $\bar{\mu} = c - 1$ and for all $\mu \in U$ solve MAX(μ) for $\mathbf{x}'_\mu$, where MAX(μ) is MIN(μ) with "max" replacing "min." Then, $\mathbf{x}^* \in \text{argmin}_{\mu \in U}(c - \boldsymbol{\mu}\mathbf{x}'_\mu)/\sqrt{\mathbf{vx}'_\mu}$. (Note that it is possible to determine which case must be considered first by solving $\max_{\mathbf{x} \in X} \boldsymbol{\mu}\mathbf{x}$ and observing whether or not the solution value exceeds c.)

The above idea is easily specialized to SKP(W). The most computationally expensive part of the modified algorithm will be the analogs of Phase 1, one where we obtain the solution value $f(w, \mu)$ by maximizing $\mathbf{vx}$ subject to $\mathbf{wx} = w$, $\boldsymbol{\mu}\mathbf{x} = \mu$ and $\mathbf{x} \in Z_+^K$, and the other where we obtain $f(w, \mu)$ by minimizing $\mathbf{vx}$ subject to the same constraints. This work will be roughly proportional to $\bar{\mu}W + (c-1)\tilde{w}$ where $\tilde{w}$ is the largest value of w for which there is no feasible $\mathbf{x}$ with $\boldsymbol{\mu}\mathbf{x} \geq c$, $\mathbf{wx} \leq w$, $\mathbf{x} \in Z_+^K$. The total work is therefore no worse than $2\bar{\mu}W$, versus the work in DPSKP which is proportional to $\bar{v}W$. For the test data set, $\bar{\mu} = \max_k \lfloor \mu_k W / w_k \rfloor = 68$ and $\bar{v} = \max_k \lfloor v_k W / w_k \rfloor = 266$. Thus, we would expect the modified algorithm to require 1/4 to 1/2 the work of DPSKP. This expectation is realized by a solution time of 0.009 seconds, excluding input.

Several final comments should be made on the basic methodology of DPSKP. The algorithm is easy to program and computer memory requirements are modest: The Steinberg-Parks problems require less than 0.1 megabytes of RAM. DPSKP is easily extended to bounded variables by solving the bounded-variable version of SKPE(W) which is is just a two-constraint, bounded-variable knapsack problem. (Dantzig [6] solves the bounded-variable knapsack problem; Nemhauser and Ullman [18] and Weingartner and Ness [26] solve multiple-constraint knapsack problems.) Furthermore, a bounded-variable algorithm could be easily modified to handle the dependent (perfectly correlated) case of SKP, problem (3).

2.2 Integer Programming Methods

The question raised and answered in this section is "Are specialized codes necessary to solve the SKP?" It is shown here that SKP(W) is readily solved using off-the-shelf integer programming tools, i.e., an algebraic modeling language and a linear-programming-based branch-and-bound solution algorithm.

Instead of hours of programming and a fraction of a second of execution time, solutions can be obtained with minutes of programming and a few seconds of execution time. All of the techniques developed are actually applicable to general problems in the form of (4) and are described as such. However, computational testing is only performed on the Steinberg-Parks problems.

2.2.1 A Simple Linearization. One of the simplest approaches to solving (4) via integer programming is to linearize the objective by taking its logarithm. The appropriate linearization depends on the sign of $c - \boldsymbol{\mu}\mathbf{x}^*$: We first solve $\bar{\mu} = \max_{\mathbf{x}\in X} \boldsymbol{\mu}\mathbf{x}$ and obtain solution $\mathbf{x}'$. By observing that $c - \boldsymbol{\mu}\mathbf{x}^*$ and $c - \bar{\mu}$ have the same sign, the problem may be separated into three cases. In case (a), $c = \bar{\mu}$ and $\mathbf{x}'$ is optimal for (4). The following discussion considers case (b) where $c > \bar{\mu}$; the linearization for case (c) where $c < \bar{\mu}$ is then a symmetric modification of case (b).

In case (b), a logarithmic linearization yields

LIN1(b)

$$
\begin{aligned}
\min_{\mathbf{h},\mathbf{d},\mathbf{x}\in X} \quad & \sum_{i=\underline{\mu}}^{\bar{\mu}} (\log(c-i))h_i - \frac{1}{2}\sum_{j=\underline{v}}^{\bar{v}} (\log j - \log(j-1))d_j \\
\text{s.t.} \quad & \sum_{i=\underline{\mu}}^{\bar{\mu}} i h_i = \boldsymbol{\mu}\mathbf{x} \\
& \sum_{i=\underline{\mu}}^{\bar{\mu}} h_i = 1 \\
& \sum_{j=\underline{v}}^{\bar{v}} d_j = \mathbf{v}\mathbf{x} \\
& h_i \in \{0,1\} \quad \text{for } i = \underline{\mu},\ldots,\bar{\mu} \\
0 \le\ & d_j \le 1 \quad \text{for } j = \underline{v}+1,\ldots,\bar{v} \\
& d_j \equiv 1 \quad \text{for } j = 1,\ldots,\underline{v}.
\end{aligned}
\tag{11}
$$

When $\boldsymbol{\mu}\mathbf{x} = i'$, $h_{i'} = 1$ and $h_i = 0$ for all $i \neq i'$, and when $\mathbf{v}\mathbf{x} = j'$, it follows that $d_j = 1$ for $j = 1,\ldots,j'$ and $d_j = 0$ for $j' > j$. Although d_j is allowed to be continuous, it will be binary in an optimal solution since $\mathbf{v}\mathbf{x}$ is integer, $-(\log j - \log(j-1))$ is an increasing function in j, and since the objective function is being minimized.

We have formulated LIN1(b) in the algebraic modeling language GAMS [1] and solved the Steinberg-Parks problems, for appropriate values of W, using the mixed-integer programming solver XA [27]. We use the same Dell laptop computer as in the previous section. The bound parameters used are $\underline{\mu} =$

W	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Total sec.	.14	.16	.19	.12	.26	.23	.26	.47	.09	.60	.36	.36	.37	.43	.13	.22

Table 1 Solution times for model LIN1(b)

$\min_k \mu_k$, $\bar{\mu} = c - 1$, $\underline{v} = \min_k v_k$, $\bar{v} = \max_k \lfloor v_k W / w_k \rfloor$. The problems had 122 variables although some of these were fixed. Table 1, lists the solution times (reported as "Resource Utilization" in the GAMS output) for the Steinberg-Parks problems for all $W \in \{3, \ldots, 18\}$ (for which $\boldsymbol{\mu}\mathbf{x}^* < c$). Tighter bounds on $\underline{v}$, $\bar{v}$, $\underline{\mu}$, and $\bar{\mu}$ can reduce the number of decision variables and speed solution time, but we pursue this issue in the next section.

The linearization for the case where $\boldsymbol{\mu}\mathbf{x}^* > c$ is analogous to LIN1(b) and is straightforward: The roles of h_i and d_j are reversed in that the h_i become continuous between 0 and 1, the d_j are binary, $d_j = 1$ implies $\mathbf{v}\mathbf{x} = j$, and $h_i = 1$ implies $\boldsymbol{\mu}\mathbf{x} \geq i$. The objective function to be linearized and maximized is $(\boldsymbol{\mu}\mathbf{x} - c)/\sqrt{\mathbf{v}\mathbf{x}}$. Initial tests with this case were not as successful as LIN1(b) at least partially because $\bar{v}$ is always larger than $\bar{\mu}$ and there are many more binary variables. Rather than trying to improve this linearization for this case, another rather different linearization is developed and tested next.

2.2.2 Another linearization. The linearization described below, for case (c) where $\boldsymbol{\mu}\mathbf{x}^* \geq c+1$, uses binary variables to enumerate all possible values for $\boldsymbol{\mu}\mathbf{x}^* \geq c+1$ and $\mathbf{v}\mathbf{x}^* \geq \underline{v}$. By solving a few auxiliary problems, the enumeration required is not burdensome, at least for the Steinberg-Parks problems. The method is described only for case (c) but with minor modifications can also be used for case (b).

For values of i and j such that $c + 1 \leq i \leq \bar{\mu}$ and $\underline{v} \leq j \leq \bar{v}$, define the binary variable y_{ij} to be 1 if $\boldsymbol{\mu}\mathbf{x} = i$ and $\mathbf{v}\mathbf{x} = j$, and to be 0 otherwise. Also, define $\rho_{ij} = (c - i)/\sqrt{j}$. Then, (4) is equivalent to

$$\text{LIN2(c)} \qquad \begin{aligned} \rho^* = \min_{\mathbf{x}\in X,\,\mathbf{y}} \quad & \sum_{(i,j)\in IJ} \rho_{ij} y_{ij} \\ \text{s.t.} \quad & \sum_{(i,j)\in IJ} i y_{ij} = \boldsymbol{\mu}\mathbf{x} \\ & \sum_{(i,j)\in IJ} j y_{ij} = \mathbf{v}\mathbf{x} \\ & \sum_{(i,j)\in IJ} y_{ij} = 1 \\ & y_{ij} \in \{0,1\} \quad \forall\,(i,j) \in IJ, \end{aligned} \tag{12}$$

where $I = \{c+1, \ldots, \bar{\mu}\}$, $J = \{\underline{v}, \ldots, \bar{v}\}$ and $IJ = I \times J$. Like the logarithmic linearization of Section 2.2.1, (12) requires the addition of only three structural constraints, but the potential number of binary variables is much larger. The required number of variables can be reduced drastically, however, by solving a sequence of auxiliary problems to find tight values for $\bar{\mu}$, $\bar{v}$, $\underline{v}$, and another bound $\bar{\rho} \geq \rho^*$. (Any elements $(i,j) \in IJ$ with $\rho_{ij} > \bar{\rho}$ are deleted.) The four-part procedure described next for solving LIN2(c) has proven successful in practice:

Step (1) Establish $\bar{\rho}$ by finding a "good" feasible solution to (4): We solve a simplification of (4) with a linear objective, $\min_{\mathbf{x}\in X'} \mathbf{s}\mathbf{x}$, to obtain $\mathbf{x}'_1$, where $X' = X \cap \{\boldsymbol{\mu}\mathbf{x} \geq c+1\}$ and $s_k = \sqrt{v_k} - \mu_k$. Then, $\bar{\rho} \equiv (c - \boldsymbol{\mu}\mathbf{x}'_1)/\sqrt{\mathbf{v}\mathbf{x}'_1}$.

Step (2) Establish $\bar{\mu}$ and $\bar{v}$: Solve $\max_{\mathbf{x}\in X} \boldsymbol{\mu}\mathbf{x}$ to obtain $\mathbf{x}'_2$ and let $\bar{\mu} = \boldsymbol{\mu}\mathbf{x}'_2$ and $\bar{v} = \mathbf{v}\mathbf{x}'_2$. The variance bound is valid since

$$\frac{c - \boldsymbol{\mu}\mathbf{x}^*}{\sqrt{\mathbf{v}\mathbf{x}^*}} \leq \frac{c - \boldsymbol{\mu}\mathbf{x}'_2}{\sqrt{\mathbf{v}\mathbf{x}'_2}} \quad \text{and} \quad c - \boldsymbol{\mu}\mathbf{x}'_2 \leq c - \boldsymbol{\mu}\mathbf{x}^* < 0 \quad \text{imply} \quad \sqrt{\mathbf{v}\mathbf{x}^*} \leq \sqrt{\mathbf{v}\mathbf{x}'_2}.$$

Additionally, if $\mathbf{x}'_2$ is a better solution to (4) than is $\mathbf{x}'_1$, $\bar{\rho}$ is reduced to $(c - \boldsymbol{\mu}\mathbf{x}'_2)/\sqrt{\mathbf{v}\mathbf{x}'_2}$.

Step (3) Establish $\underline{v}$: Solve $\min_{\mathbf{x}\in X'} \mathbf{v}\mathbf{x}$ to obtain $\mathbf{x}'_3$, where $X' = X \cap \{\boldsymbol{\mu}\mathbf{x} \geq c+1\}$. Let $\underline{v} = \mathbf{v}\mathbf{x}'_3$ and update $\bar{\rho}$ if $\mathbf{x}'_3$ is a better solution for (4) than are $\mathbf{x}'_1$ and $\mathbf{x}'_2$.

Step (4) Solve LIN2(c): After the three auxiliary problems are solved and good values for $\bar{\mu}$, $\bar{v}$, $\underline{v}$ and $\underline{\rho}$ are established, a "tight" version of LIN2(c) is then solved.

W	19	20	21	22	23	24	25	26	27	28	29	30
Step (1) sec.	.07	.08	.08	.06	.08	.06	.07	.08	.09	.07	.08	.08
Step (2) sec.	.08	.08	.04	.08	.09	.07	.07	.08	.09	.05	.08	.11
Step (3) sec.	.08	.06	.07	.08	.10	.08	.09	.08	.08	.06	.06	.07
LIN2(c) sec.	.08	.11	.06	.05	.09	.08	.11	.17	.21	.10	.20	.23
Total sec.	.31	.33	.25	.27	.36	.29	.34	.41	.47	.28	.42	.49

Table 2 Solution statistics for LIN2(c) and auxiliary problems.

The four-part procedure described above was tested on the Steinberg-Parks problems for $W \in \{19, \ldots, 30\}$ for which $\rho^* < 0$. Table 2 displays the solution times of the individual auxiliary problems and LIN2(c) for each relevant value of W.

The auxiliary problems did make a significant difference in problem size and solution time for LIN2(c). LIN2(c) contains from 13 to 316 variables as solved, and total solution time never exceeds one half second. When we try to solve LIN2(c) without the auxiliary problems (using more easily calculated bounds), problems sizes range from 250 to 1824 variables and some run times exceed 30 seconds.

So, the IP approach yields solutions reasonably quickly and the programming effort is minimal even though a number of auxiliary problems may need to be solved. The approach does not really depend on the form of the model's constraints, so it is much more flexible than DP. However, both the IP and DP approaches require that returns be independent normal random variables. General return distributions with an arbitrary dependency structure are allowed in the Monte Carlo method we develop in the rest of the paper.

3 SKP WITH GENERAL RETURN DISTRIBUTIONS

In this section, we consider (1), which for convenience we restate here as

$$\begin{aligned} z^* = \quad & \max_{\mathbf{x}} \quad P\left(\mathbf{rx} \geq c\right) \\ & \text{s.t.} \quad \mathbf{x} \in X, \end{aligned} \tag{1}$$

where $\mathbf{r}$ is a random vector with a general distribution. Thus, $\mathbf{r}$ may be non-normal and may have dependent components. In the context of the stochastic knapsack problem with returns that are independent both between and within item types, (1) specializes to (2) with r_{kl}, $k = 1, \ldots, K$, $l \in \mathcal{L}_k$, independent.

And, when returns are perfectly correlated within an item type but independent between types, (1) specializes to (3) with r_k, $k = 1, \ldots, K$, independent. We will consider these two special cases in our computational work, but we develop the Monte Carlo solution procedure in the more general context of (1), without independence assumptions on the components of $\mathbf{r}$.

When stochastic optimization problems such as (1) do not have a special structure such as normally distributed returns (see Section 2), it is usually necessary to resort to approximation procedures in order to solve the problem, approximately. One common approach is to replace the "true" distribution of the random vector $\mathbf{r}$ with an approximating distribution that is more manageable from a computational perspective; see Wets [25, §6]. A Monte Carlo procedure that generates independent and identically distributed (i.i.d.) observations, $\mathbf{r}^j$, $j = 1, \ldots, m$, from the distribution of $\mathbf{r}$ may be viewed from this perspective: These observations (which we will also refer to as *scenarios*) are the realizations of an m-point empirical approximating distribution. As we will show, modest values of m can yield computationally tractable optimization models that provide good approximations of SKP.

Let $I(\cdot)$ be the indicator function that takes on the value 1 if its argument is true, and is 0 otherwise. With this notation,

$$P(\mathbf{rx} \geq c) = EI(\mathbf{rx} \geq c) = E\left[\frac{1}{m}\sum_{j=1}^{m} I\left(\mathbf{r}^j\mathbf{x} \geq c\right)\right].$$

Thus, the approximating problem based on an empirical distribution is

$$\begin{aligned} U_m = \quad & \max_{\mathbf{x}} \quad \frac{1}{m}\sum_{j=1}^{m} I\left(\mathbf{r}^j\mathbf{x} \geq c\right) \\ & \text{s.t.} \quad \mathbf{x} \in X. \end{aligned} \tag{13}$$

By observing that

$$\begin{aligned} z^* = \max_{\mathbf{x}\in X} P(\mathbf{rx} \geq c) \quad &= \max_{\mathbf{x}\in X} E\left[\tfrac{1}{m}\textstyle\sum_{j=1}^{m} I\left(\mathbf{r}^j\mathbf{x} \geq c\right)\right] \\ &\leq E\left[\max_{\mathbf{x}\in X} \tfrac{1}{m}\textstyle\sum_{j=1}^{m} I\left(\mathbf{r}^j\mathbf{x} \geq c\right)\right] = EU_m, \end{aligned} \tag{14}$$

we see that U_m is an upper bound, in expectation, on the optimal solution value z^*; see Mak *et al.* [14].

Estimates of EU_m are valuable in ascertaining the quality of a feasible candidate solution $\hat{\mathbf{x}} \in X$. We may estimate the objective value, $P(\mathbf{r}\hat{\mathbf{x}} \geq c)$, via

$$L_m = \frac{1}{m}\sum_{j=1}^{m} I(\mathbf{r}^j\hat{\mathbf{x}} \geq c).$$

Because $\hat{\mathbf{x}}$ is, in general, suboptimal, $EL_m = P(\mathbf{r}\hat{\mathbf{x}} \geq c) \leq z^*$. As we show below, estimates of the upper bound EU_m can be used to bound the optimality gap, $z^* - P(\mathbf{r}\hat{\mathbf{x}} \geq c)$.

We generate a candidate solution $\hat{\mathbf{x}} \in X$ by solving a single approximating problem of the form (13). It is clearly desirable to ascertain the quality of such a solution, and to do so we follow Mak *et al.* [14]. This procedure consists of using the method of batch means to construct a one-sided confidence interval of the optimality gap, $z^* - P(\mathbf{r}\hat{\mathbf{x}} \geq c)$, by forming i.i.d. observations of

$$G_m = U_m - L_m = \max_{\mathbf{x} \in X} \left[\frac{1}{m} \sum_{j=1}^{m} I\left(\mathbf{r}^j \mathbf{x} \geq c\right) \right] - \frac{1}{m} \sum_{j=1}^{m} I\left(\mathbf{r}^j \hat{\mathbf{x}} \geq c\right).$$

Since $EU_m \geq z^*$ and $EL_m = P(\mathbf{r}\hat{\mathbf{x}} \geq c)$, it follows that $EG_m \geq z^* - P(\mathbf{r}\hat{\mathbf{x}} \geq c)$. Hence, we may use multiple observations of G_m to construct point and interval estimates for the optimality gap.

The upper and lower bound estimators that define G_m use the same stream of random numbers $\mathbf{r}^j$, $j = 1, \ldots, m$; this use of *common random numbers* is a well-known variance reduction technique. (See, for example, Law and Kelton [13, §11.2] for a general discussion of common random numbers; for computational results in stochastic programming, see Mak *et al.* [14].) In our current setting, common random numbers have the additional benefit of ensuring nonnegative estimates of the optimality gap, since, by construction $G_m \geq 0$; this could not be guaranteed if U_m and L_m were estimated separately with distinct random number streams. Before summarizing our Monte Carlo procedure for approximately solving SKP, we turn to the issue of evaluating L_m and U_m.

Evaluating L_m is straightforward: Given $\hat{\mathbf{x}}$, we generate $\mathbf{r}^j$, $j = 1, \ldots, m$, and for each observation simply test whether or not $\mathbf{r}^j\hat{\mathbf{x}} \geq c$ and compute $L_m = \frac{1}{m}\sum_{j=1}^{m} I\left(\mathbf{r}^j\hat{\mathbf{x}} \geq c\right)$.

To calculate U_m, we convert (13) into the following equivalent IP

$$\begin{aligned} U_m = \quad \max_{\mathbf{x},\mathbf{y}} \quad & \frac{1}{m}\sum_{j=1}^{m} y_j & & \\ \text{s.t.} \quad & \mathbf{x} \in X & & \\ & \mathbf{r}^j\mathbf{x} \geq c\, y_j - M_j(1-y_j) & \forall\, j = 1, \ldots, m & \\ & \mathbf{y} \in \{0,1\}^m. & & \end{aligned} \tag{15}$$

Here, $M_j > 0$ is large enough to ensure that $\mathbf{r}^j\mathbf{x} \geq c\, y_j - M_j(1-y_j)$ is a vacuous constraint when $y_j = 0$.

The Monte Carlo Procedure for solving SKP begins by solving an empirical approximating problem (15) with m' scenarios to generate a candidate solution

$\hat{\mathbf{x}}$. Then, we use the method of common random numbers, with a batch size of m, to construct an approximate $(1-\alpha)$-level confidence interval on the optimality gap, $z^* - P(\mathbf{r}\hat{\mathbf{x}} \geq c)$. In practice, we typically choose m' larger than m in an attempt to find a good candidate solution.

Procedure MCSKP
Input: Data for SKP with K items: $\mathbf{w}$, c, W, and distribution for $\mathbf{r}$. Batch size m, sample size (number of batches) n, and size of approximating problem to generate candidate solution, m'. Confidence interval level $1-\alpha$ and z_α satisfying $P(N(0,1) \leq z_\alpha) = 1-\alpha$.
Output: Solution $\hat{\mathbf{x}}$, approximate $(1-\alpha)$-level confidence interval $[0, \bar{G}(n)+\epsilon_G]$ on the optimality gap.

{

/* Generate Candidate Solution */

Generate $\mathbf{r}^1, \ldots, \mathbf{r}^{m'}$ i.i.d. from the distribution of $\mathbf{r}$;

$\hat{\mathbf{x}} \leftarrow \underset{\mathbf{x} \in X}{\operatorname{argmax}} \left[\frac{1}{m'} \sum_{j=1}^{m'} I\left(\mathbf{r}^j \mathbf{x} \geq c\right)\right]$;

/* Optimality Gap Calculations */

For ($i = 1$ to n) {

Generate $\mathbf{r}^{i1}, \ldots, \mathbf{r}^{im}$ i.i.d. from the distribution of $\mathbf{r}$;

$G_m^i \leftarrow \max_{\mathbf{x} \in X} \left[\frac{1}{m} \sum_{j=1}^{m} I\left(\mathbf{r}^{ij} \mathbf{x} \geq c\right)\right] - \frac{1}{m} \sum_{j=1}^{m} I\left(\mathbf{r}^{ij} \hat{\mathbf{x}} \geq c\right)$;

}

$\bar{G}(n) \leftarrow \frac{1}{n} \sum_{i=1}^{n} G_m^i$;

$S_G^2(n) \leftarrow \frac{1}{n-1} \sum_{i=1}^{n} \left[G_m^i - \bar{G}(n)\right]^2$;

$\epsilon_G \leftarrow z_\alpha S_G(n)/\sqrt{n}$;

Print{"Approximate solution to SKP:",$\hat{\mathbf{x}}$};

Print{"Confidence interval on the optimality gap:",$[0, \bar{G}(n) + \epsilon_G]$};

}

The MCSKP procedure was implemented in GAMS [1] and the IPs solved using CPLEX Version 3.0 [3]. All computational tests in this section were performed on an IBM RS-6000 Model 590 computer with 512 megabytes of RAM. Because we already know optimal solutions to the Steinberg-Parks problems, and can perform exact evaluations of $P(\mathbf{r}\hat{\mathbf{x}} \geq c)$ for candidate solutions $\hat{\mathbf{x}}$,

W	$\bar{G}(n)$	ϵ_G	95% CI	$P(\mathbf{r}\hat{\mathbf{x}} \geq c)$	z^*	CPU (min.)
10	0.006	0.003	[0,0.009]	0.014	0.014	19.4
15	0.072	0.010	[0,0.082]	0.124	0.173	26.9
20	0.052	0.010	[0,0.062]	0.549	0.588	32.7
25	0.020	0.007	[0,0.027]	0.915	0.915	25.6
30	0.025	0.005	[0,0.030]	0.978	0.995	19.7

Table 3 Results of the Monte Carlo solution procedure for the Steinberg-Parks SKPs. Returns are normal random variables that independent between and within item types. In these computations $m' = 200$ (candidate generation), $m = 100$ (batch size), and $n = 30$ (number of batches).

we can make some interesting observations regarding the performance of the Monte Carlo solution procedure from Table 3. In two of the five cases, the $\hat{\mathbf{x}}$ found by solving the empirical problem with $m' = 200$ scenarios is optimal. By definition, the approximate 95% confidence interval achieves the desired *coverage* provided that $z^* - P(\mathbf{r}\hat{\mathbf{x}} \geq c)$ falls within the interval. For example, when $W = 20$, $z^* - P(\mathbf{r}\hat{\mathbf{x}} \geq c) = 0.039$ falls in $[0, 0.062]$. Table 3 indicates that the desired coverage is achieved in each of the five cases. In fact, in each case the optimality gap is smaller than the point estimate $\bar{G}(n)$; this is not surprising since $E\bar{G}(n) \geq z^* - P(\mathbf{r}\hat{\mathbf{x}} \geq c)$. Because the point estimate of the gap is biased in this manner, we tend to obtain conservative confidence interval statements (a caveat to this, due in part to the discrete nature of the integer SKPs, is discussed below). We note that when $W = 30$, the confidence interval provides an effectively vacuous statement since the probability of achieving the target is within 0.03 of 1. (The MCSKP procedure must be applied with some care, if at all, when $P(\mathbf{r}\hat{\mathbf{x}} \geq c)$ is close to 0 or 1.)

The primary goal of the MCSKP procedure is to obtain a solution $\hat{\mathbf{x}}$ of "high quality" and to make a probabilistic statement concerning this quality. The procedure does not include a point estimate of $P(\mathbf{r}\hat{\mathbf{x}} \geq c)$ because we regard this of secondary importance relative to obtaining an $\hat{\mathbf{x}}$ of high quality. Of course, a point estimate is straightforward to compute, if desired.

In order to study the effect of the number of scenarios m' on the quality of the candidate solution, $\hat{\mathbf{x}}$, we took the problem with the poorest solution (widest optimality gap) from Table 3 ($W = 15$) and ran the Monte Carlo procedure for various values of m'. The results are summarized in Table 4. To reduce the variability due to sampling, the candidate-generation and optimality-gap-estimation phases of the MCSKP procedure were, respectively, initialized with

m'	$\bar{G}(n)$	ϵ_G	95% CI	$P(\mathbf{r}\hat{\mathbf{x}} \geq c)$	CPU (min.)
50	0.091	0.010	[0,0.101]	0.102	25.8
100	0.091	0.010	[0,0.101]	0.102	26.0
200	0.072	0.010	[0,0.082]	0.124	26.9
300	0.040	0.010	[0,0.051]	0.159	28.2
400	0.026	0.009	[0,0.035]	0.173	32.1
500	0.026	0.009	[0,0.035]	0.173	36.8
600	0.026	0.009	[0,0.035]	0.173	43.4

Table 4 We illustrate the quality of the candidate solution generated by solving empirical approximating problems for the SKP with $W = 15$ for various batch sizes m'. This problem has $z^* = 0.173$. For constructing the confidence intervals we use $m = 100$ and $n = 30$. The CPU times are for the entire MCSKP procedure.

the same seeds for generating pseudo-random variates for each value of m'. This has two effects: First, when increasing m' from, say, 300 to 400 we have simply added 100 additional scenarios to the original 300. Second, when the candidate-generation phase finds the same $\hat{\mathbf{x}}$ for different values of m' (i.e., $m' = 50, 100$ and $m' = 400, 500, 600$) the gap-estimation results are identical. Note that $m' = 400, 500$, and 600 all yield an optimal solution.

As Tables 3 and 4 indicate, even when the candidate-generation phase finds an optimal solution, we still obtain confidence intervals with widths ranging from 0.009 to 0.035. There are two reasons for this: First, there is a contribution due to $\bar{G}(n)$ that originates from the inequality in (14), obtained by exchanging the optimization and expectation operators. Second, there is a contribution due to sampling error which is captured in ϵ_G. Table 5 shows a decrease in both these terms as the batch size m grows. In fact, it is possible to show that EU_m decreases monotonically in m [17, 14]. The increase in CPU times with larger batch sizes in Table 5 (and to a lesser extent in Table 4) is due, in part, to the IP (15) becoming larger. But, the IP optimality gap must be shrunk to a value less than $1/m$ to ensure optimality, and this also results in increasing times.

As indicated in Section 1, certain systems lead to SKPs in which the returns within (as well as between) item types are not independent. Table 6 summarizes computational results for a variant of the Steinberg-Parks problems in which the returns are normally distributed and independent between item types but are perfectly correlated within each type. Because the number of integer variables in (15) is significantly smaller than for the independent case, the computational effort is significantly less for this model.

m	$\bar{G}(n)$	ϵ_G	95% CI	CPU (min.)
25	0.075	0.020	[0,0.095]	24.6
50	0.056	0.012	[0,0.068]	26.1
100	0.026	0.009	[0,0.035]	32.1
200	0.019	0.006	[0,0.025]	41.7
300	0.010	0.004	[0,0.014]	115.2
400	0.008	0.003	[0,0.011]	208.4

Table 5 We illustrate the effect of the batch size m on the tightness of the confidence interval by applying the MCSKP procedure to the SKP with $W = 15$ for an optimal candidate solution. We use a sample size of $n = 30$. The CPU times include the time required to solve the $m' = 400$ scenario problem to find an optimal candidate solution.

W	$\bar{G}(n)$	ϵ_G	95% CI	$P(\mathbf{r}\hat{\mathbf{x}} \geq c)$	CPU (min.)
10	0.000	0.000	[0,0.000]	0.090	3.2
15	0.000	0.001	[0,0.001]	0.327	6.2
20	0.021	0.007	[0,0.028]	0.561	9.2
25	0.017	0.006	[0,0.023]	0.872	5.7
30	0.016	0.005	[0,0.021]	0.973	2.8

Table 6 Results of the Monte Carlo solution procedure for SKPs with normal returns that are independent between, but perfectly correlated within item types. In these computations $m' = 200$ (candidate generation), $m = 100$ (batch size), and $n = 30$ (number of batches).

Note that in Table 6 the confidence interval width is actually 0 for $W = 10$ and is 0.001 for $W = 15$. While this may be somewhat disconcerting, when $W = 10$ each of the $n = 30$ empirical problems ($m = 100$) yielded the same solution $\hat{\mathbf{x}}$ as the candidate-generation problem ($m' = 200$). And, when $W = 15$, 29 of the 30 empirical problems generated the same solution $\hat{\mathbf{x}}$ as the candidate-generation problem (to four digits; $\bar{G}(n) = 0.0003$ for this case). Such results are partly due to the discrete nature of the integer SKP and would be less likely to occur if the decision variables were continuous, particularly if the solutions were not extreme points of X.

Finally, Table 7 summarizes the computational results for another variant of the Steinberg-Parks problems in which distributions of the returns are assumed

W	$\bar{G}(n)$	ϵ_G	95% CI	CPU (min.)
10	0.003	0.002	[0,0.005]	19.3
15	0.030	0.011	[0,0.041]	26.1
20	0.057	0.015	[0,0.072]	32.4
25	0.020	0.007	[0,0.027]	26.3
30	0.014	0.004	[0,0.018]	19.4

Table 7 Results of the Monte Carlo solution procedure for SKPs with uniformly distributed returns that are independent between and within item types. In these computations $m' = 200$ (candidate generation), $m = 100$ (batch size), and $n = 30$ (number of batches).

to be uniform, having the same mean and variance as the normal distributions of the original Steinberg-Parks data. Here, the returns are independent between and within item types. In this case, both the required computational effort and the magnitude of the confidence interval widths are very similar to that for normally distributed returns (see Table 3).

4 CONCLUSIONS

This paper has considered stochastic integer programming problems, with deterministic constraints, where the objective is to maximize the probability of meeting or exceeding a certain return threshold. We have developed three solution procedures. In Section 2.1, we presented a new dynamic-programming method for the special case of the stochastic knapsack problem with normally distributed returns that are independent between and within item types. This method is conceptually simple, easy to program, easy to modify for bounded variables, and significantly faster than previously available procedures. In Section 2.2, we described integer programming techniques with the same structure on the random returns but with more general constraint sets. We used two different linearized integer programs coupled with several auxiliary integer programs. These methods were tested and shown to be effective. Finally, the Monte Carlo solution procedure of Section 3 addressed problems under very general assumptions regarding the distribution of the vector of random returns. Due to the more general problem structure, we solved an approximating problem whose solution quality was specified only in a probabilistic sense. Nevertheless, our computational results demonstrated that good solutions can be obtained with modest sample sizes.

Acknowledgments

Kevin Wood thanks the Office of Naval Research and the Air Force Office of Scientific Research for its support with this research. David Morton's research was supported by the National Science Foundation through grant DMI-9702217.

References

[1] Brooke, A., Kendrick, D., and Meeraus, A., *GAMS: A User's Guide*, The Scientific Press, San Francisco (1992).

[2] Carraway, R.L., Morin, T.L., and Moskowitz, H., "Generalized Dynamic Programming for Stochastic Combinatorial Optimization," *Operations Research*, **37**, 819-829 (1989).

[3] CPLEX Manual, *Using the CPLEX^{TM} Callable Library and CPLEX^{TM} Mixed Integer Library,* CPLEX Optimization, Inc., Incline Village, Nevada, 1993.

[4] Carraway, R.L., Schmidt, R.L., and Weatherford, L.R., "An Algorithm for Maximizing Target Achievement in the Stochastic Knapsack Problem with Normal Returns," *Naval Research Logistics*, **40**, 161-173 (1993).

[5] Chiu, S.Y., Lu, L., and Cox, L.A., "Optimal Access Control for Broadband Services: Stochastic Knapsack with Advance Information," *European Journal of Operations Research*, **89**, 127-134 (1996).

[6] Dantzig, G.B., "Discrete-Variable Extremum Problems," *Operations Research*, **5**, 266-277 (1957).

[7] Dreyfus, S.E. and Law, M.L., *The Art and Theory of Dynamic Programming*, Academic Press, New York (1977).

[8] Gavious, A. and Rosberg, Z., "A Restricted Complete Sharing Policy for a Stochastic Knapsack Problem in B-ISDN," *IEEE Transactions on Communications*, **42**, 2375-2379 (1994).

[9] Geoffrion, A.M., "Solving Bicriterion Mathematical Programs," *Operations Research*, **15**, 39-54 (1967).

[10] Greenberg, H.J., "Dynamic Programming with Linear Uncertainty," *Operations Research*, **16**, 675-678 (1968).

[11] Henig, M.I., "Risk Criteria in the Stochastic Knapsack Problem," *Operations Research*, **38**, 820-825 (1990).

[12] Ishii, H. and Nishida, T., "Stochastic Linear Knapsack Problem: Probability Maximization Model," *Mathematica Japonica*, **29**, 273-281 (1984).

[13] Law, A.M. and Kelton, W.D., *Simulation Modeling and Analysis*, McGraw-Hill, New York (1991).

[14] Mak, W.K., Morton, D.P., and Wood, R.K., "Monte Carlo Bounding Techniques for Determining Solution Quality in Stochastic Programs," Technical Report, The University of Texas at Austin (1997).

[15] Marchetti-Spaccamela, A. and Vercellis, C., "Stochastic On-Line Knapsack Problems," *Mathematical Programming*, **68**, 73-104 (1995).

[16] Morita, H., Ishii, H., and Nishida, T., "Stochastic Linear Knapsack Programming Problem and Its Application to a Portfolio Selection Problems," *European Journal of Operations Research*, **40**, 329-336 (1989).

[17] Norkin, V.I., Pflug, G.Ch., and Ruszczyński, A., "A Branch and Bound Method for Stochastic Global Optimization," Working Paper, IIASA (1996).

[18] Nemhauser, G.L., and Ullman, Z., "Discrete Dynamic Programming and Capital Allocation," *Management Science*, **15**, 494-505 (1969).

[19] Papastavrou, J.D., Rajagopalan, S., Kleywegt, A.J., "Discrete Dynamic Programming and Capital Allocation," *Management Science*, **42**, 1706-1718 (1996).

[20] Prékopa, A., *Stochastic Programming*, Kluwer Academic Publishers, Dordrecht (1995).

[21] Ross, K.W., *Multiservice Loss Models for Broadband Telecommunication Networks*, Springer-Verlag, London (1995).

[22] Ross, K.W. and Tsang, D.H.K., "The Stochastic Knapsack Problem," *IEEE Transactions on Communications*, **37**, 740-747 (1989).

[23] Sniedovich, M. "Preference Order Stochastic Knapsack Problems: Methodological Issues," *Journal of the Operational Research Society*, **31**, 1025-1032 (1980).

[24] Steinberg, E., and Parks, M.S., "A Preference Order Dynamic Program for a Knapsack Problem with Stochastic Rewards," *Journal of the Operational Research Society*, **30**, 141-147 (1979).

[25] Wets, R.J.-B. (1989): Stochastic Programming, in G.L. Nemhauser, A.H.G. Rinnooy Kan, and M.J. Todd (eds.) *Handbooks in Operations Research and Management Science*, Elsevier Science Publishers, Amsterdam.

[26] Weingartner, M.H., and Ness, D.N., "Methods for the Solution of Multidimensional 0/1 Knapsack Problems", *Operations Research*, **15**, 83-103 (1967).

[27] XA, Professional Linear Programming System, Version 2.2, Sunset Software Technology, San Marino, California (1993).